AMENDMENTS TO THE SPECIFICATION

Please amend paragraph [0002] as follows:

[0002] The subject matter of this application is related to the subject matter in a co-pending non-provisional application by the same inventors as the instant application and filed on the same day as the instant application entitled, "Method and Apparatus to Facilitate Sharing Instruction Code in a Multitasking Virtual Machine," having serial number ~~TO BE ASSIGNED~~10/043,801, and filing date ~~TO BE ASSIGNED~~ January 10, 2002 ~~(Attorney Docket No. SUN-P6119-RSH)~~.

Please amend paragraph [0006] as follows:

[0006] Virtual machines for object-oriented programming languages with dynamic class loading typically load the code of a class when a program resolves a symbolic reference to that class for the first time. The class needs to be initialized subsequently when the program uses it for the first time. Loading and initialization of a class are two separate events. Initialization of a class may never take place even though the class has been loaded before. In the case of the Java programming language, the initialization of a class consists of executing some code, known as the class's static initializer, that brings the class's variables (also known as the static variables) to a well-defined initialized state. A virtual machine implementation may choose to set a class to the initialized state upon its loading when no action is required to initialize that class. For instance, in the Java programming language, no action is required to initialize a class when this class has no declared static initialization sequence, and either no non-final static variables, or non-final static variables that are all declared to be set to a default value. In this case, a virtual machine implementation can ~~benefits~~ benefit from setting such initialization-less classes to the initialized state upon class loading.